NLUI Server User's Guide

By Vadim Berman

Monday, 19 March 2012

Overview

NLUI (Natural Language User Interface) Server is designed to run scripted applications driven by natural language interaction. Just like a web server application serves web pages, through which the user interacts in structured manner with a set of interactive controls, NLUI Server scripts query and obtain information from the user via a freely flowing natural linguistic interaction, without forcing the user to go through a rigid dialogue tree.

NLUI Server is based on Carabao Language Kit runtime, which converts unstructured textual data into a series of references to entities in a semantic network, tagged with extensive metadata.

GUI vs. NLUI: Basic Concepts

Let us refer to GUI as an example of interaction to explain how NLUI works. A typical GUI application contains a series of screens, presenting various prompts to the user and returning the user's response. The prompts may be of different types: a text box, a check box when a toggle is required, a set of radio buttons or a combo box when the selection must be done from a fixed set. From the conceptual point of view, there are three levels:

1. Application. Depending on the platform, the application can take form of a website, an app in a smart phone, or a native desktop OS application.
2. Screen (may be called page or dialog or window). A screen represents a stage in processing, in the process of which the data is obtained from the user.
3. Prompt (usually called control). A prompt holds one single piece of data obtained from the user, whether it is a piece of text or a fixed selection.

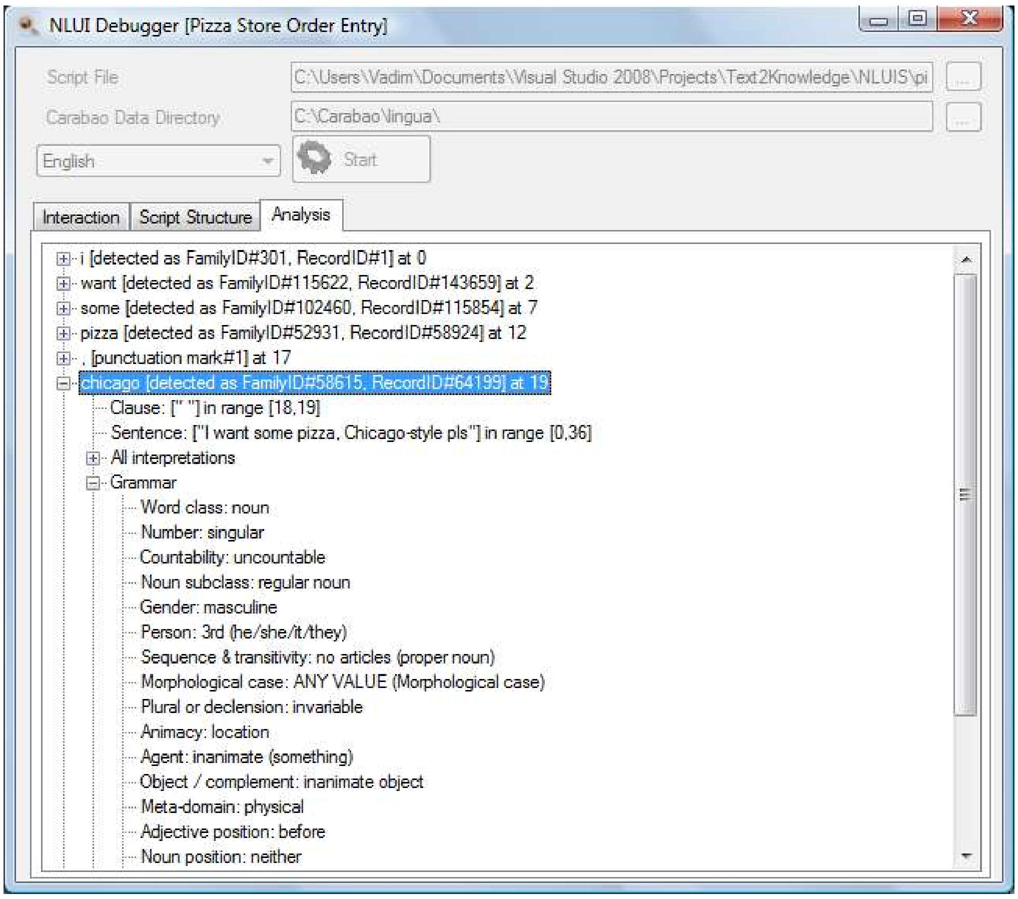
It does not matter what model of interaction is adopted specifically, whether it is MVC, wizard, etc. The common part is that graphical interface is essentially a series of information exchange steps between the user and the program.

NLUI Server uses a similar model. The **service** (which is associated with an instance of an actual service of NLUI Server) is an equivalent of a web server in the web world (or a set of apps in a smart phone, Programs menu in Windows, etc.). The service contains one or more supported **domain**s, equivalent to an "application" in the GUI model. Each **domain**, in turn, consists of a series of **stage**s, while each stage may have to obtain several pieces of data from the user.

This is where NLUI somewhat diverges from GUI. While GUI controls make sure that the user's response has a high degree of conformance with the output specifications, natural language input is completely free. The text may contain relevant or irrelevant information. The rigid classic dialogue systems, which make sure every single prompt is answered in a strict order as defined, may not always be able to do the job. It is too easy to throw this kind of system off the scent, even when the user is willing to cooperate. NLUI Server takes a different approach. Every input is scanned by multiple invisible "prompts" to see whether there is information for these prompts. This way, the user does not have to answer every question separately and can skip ahead by providing the required information at once. As the user may not be even prompted about this data, in NLUI Server parlance the nodes are called **obtain** nodes. To summarize the logical structure analogous to the GUI model:

1. Domain.
2. Stage.
3. Obtain node.

Isn't it too cumbersome to make all the nodes look at the data? This is where Carabao runtime works its magic, reducing the linguistic input to a series of structured codes. The results of the analysis can be seen in the *Analysis* tab of NLUI Debugger as shown on the illustration below:



Script File: C:\Users\Vadim\Documents\Visual Studio 2008\Projects\Text2Knowledge\NLUIS\pi

Carabao Data Directory: C:\Carabao\lingua\

English

```
⊞ i [detected as FamilyID#301, RecordID#1] at 0
⊞ want [detected as FamilyID#115622, RecordID#143659] at 2
⊞ some [detected as FamilyID#102460, RecordID#115854] at 7
⊞ pizza [detected as FamilyID#52931, RecordID#58924] at 12
⊞ , [punctuation mark#1] at 17
⊟ chicago [detected as FamilyID#58615, RecordID#64199] at 19
    Clause: [" "] in range [18,19]
    Sentence: ["I want some pizza, Chicago-style pls"] in range [0,36]
  ⊞ All interpretations
  ⊟ Grammar
      Word class: noun
      Number: singular
      Countability: uncountable
      Noun subclass: regular noun
      Gender: masculine
      Person: 3rd (he/she/it/they)
      Sequence & transitivity: no articles (proper noun)
      Morphological case: ANY VALUE (Morphological case)
      Plural or declension: invariable
      Animacy: location
      Agent: inanimate (something)
      Object / complement: inanimate object
      Meta-domain: physical
      Adjective position: before
      Noun position: neither
```

Natural Language Entities in Carabao

While the complete description of what Carabao does and how may be too bulky to fit in here, there is a small set of basic concepts essential for understanding how to work with Carabao-based applications.

Words in Carabao are grouped into **families**. A family is a set of synonyms, alternative spellings, and inflected forms related to a specific concept. For example, the words *physician* and *doctor* refer to the same concept: "*a licensed medical practitioner*". Therefore, they both belong to the same family.

It is important to note that the family ID numbers are the same across languages. For instance, the family referring to "*a licensed medical practitioner*" is designated the number 63787, in all supported languages.

As families belong to a crosslingual semantic network, a family may be connected to a family which may be its sub-concept (hyponym) or its super-concept (hypernym). For example, *pediatrician* is a hyp**o**nym of *physician*, or, inversely, *physician* is a hyp**er**nym of *pediatrician*. As NLUI Server natively supports looking up a **hypernym**, it is easy to look for, say, all kinds of Asian food (Chinese, Japanese, Indian, Korean, etc.) by simply setting the condition *hypernym=8981*.

In a somewhat confusing manner, Carabao long used the term **domain** to point to a subject area in which the terms are grouped. Please note that a Carabao domain is not the same as the domain tag in an NLUI Server script. Carabao domain is a topic. For example, doctors, diseases, medical equipment concepts belong to the domain of medicine. NLUI Server natively supports looking up a domain. For example, a concept like *Saturn* (the planet) is associated with the domain "outer space" and criterion like *domain=56014* will be satisfied for that concept.

Of course, words are ambiguous. Carabao is used to determine the actual meaning used in the input. While the accuracy of results is relatively high, natural language processing is not always reliable; unless the user's choices are very broad, it is recommended to leave the *detectedEntitiesOnly* setting to default (*false*). If *detectedEntitiesOnly* is set to *true*, then only interpretations singled out by Carabao will be considered.

Please refer to Carabao User's Manual for more information on Carabao and how to view and edit the lexical data.

Variables and Executable Code

NLUI Server relies on .NET for powerful scripting capabilities. The variables in the script refer directly to the data types in .NET runtime, not necessarily system types or those in the attached source code file. In fact, any library and any type can be used, either to store data, or to invoke arbitrary code. This essentially means that all the tens of languages which can compile into .NET CLR, are supported.

Installation and Running

The installation package includes the following:

1. The shared .NET assembly and Carabao runtime grouped with NLUI Debugger and Carabao data management tools. This is an essential component. Please refer to Carabao User's Guide on how to work with Carabao Data Manager.
2. NLUI Server application, which is installed as a Windows service. This must be selected only when installing on the server machine.
3. Carabao lexical data. These are plain data files, which do not have to be registered or mounted, deployment is as simply as copying. They are required for both NLUI Debugger and NLUI Server to function.

Running NLUI Server

NLUI Server is a standard Windows service. Upon installation, it is made to start automatically when the machine starts. Modifications can be made via the *Services* console, and the settings may be modified by editing *svcNLUIS.exe.config* file in the installation directory. The **service** configuration file must refer to a **script** configuration file, which is loaded on the service startup.

The service configuration file is a standard .NET configuration. Please refer to the extensive Microsoft documentation on *Configuration File Schema for the .NET Framework* (current URL: http://msdn.microsoft.com/en-us/library/1fk1t1t0.aspx). Normally, the sections of interest are editable by *SvcConfigEditor.exe* tool provided by Microsoft.

Script Development

Obligatory "Hello, world" Example

Let us start with a simple example: a script that asks the user to say "hello" in whatever way the user likes, and repeats it.

The example only contains one variable, an introduction message, one domain, one stage, and one obtain element:

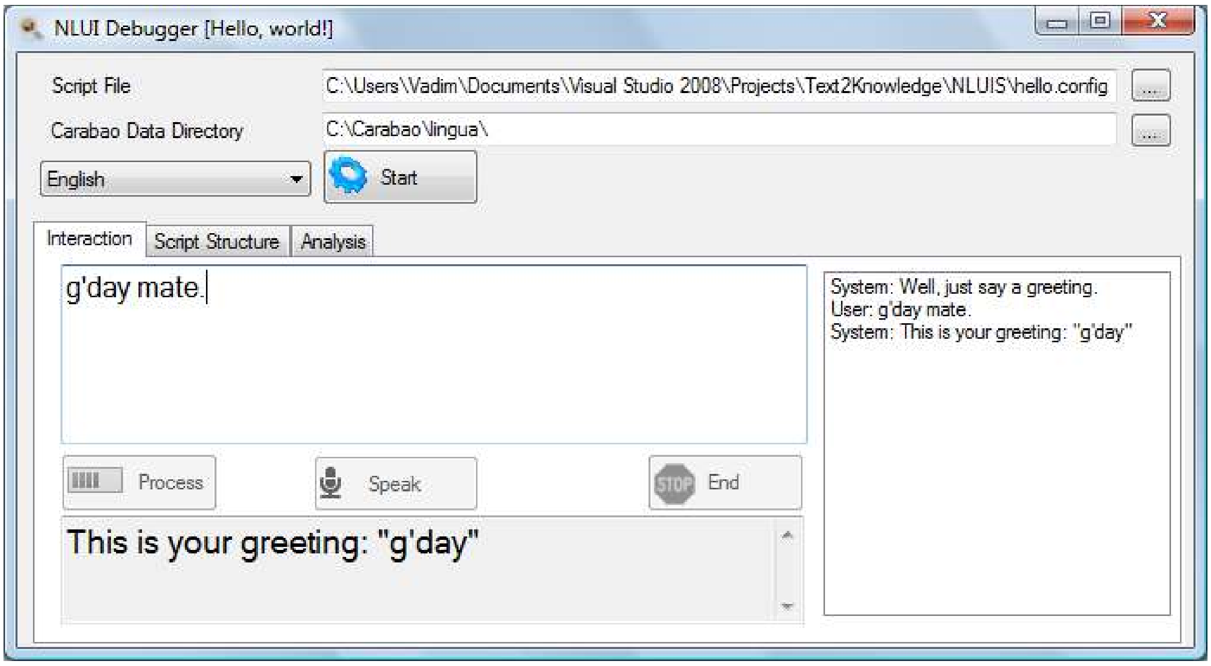
```
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://www.linguasys.com/nlui" name="Hello, world!"
timeoutInMinutes="1" defaultLanguageCode="en" languageMap="en=ENG,"
lexiconDataDir="E:\Carabao\" messageConcatenator=", ">
  <variables>
    <variable name="usersHello" type="System.String" />
  </variables>
  <introductionMessage>
    <message lang="en">
      <template>Well, just say a greeting.</template>
    </message>
  </introductionMessage>
  <domain name="helloWorld">
    <stage maxAttemptExceededAction="abort" name="firstAndOnlyStage" >
      <!-- The only stage: gather the information, and send back to the user the
matching entity -->
      <obtain targetVariable="usersHello" entityCondition="hypernym=45708;"
extractionMethod="matchingEntityOnly" maxExtractedEntityCount="1" maxAttempts="5">
```

```
        <prompt>
          <message lang="en">
            <template>Just say hello. In any way you like.</template>
          </message>
        </prompt>
      </obtain>
      <baseFinishMessage>
        <message lang="en">
          <template>This is your greeting: "{0}"</template>
        </message>
        <arguments>
          <variableName>usersHello</variableName>
        </arguments>
      </baseFinishMessage>
    </stage>
  </domain>
</service>
```

Let's look at the important parts of the code:

- A *variable* "usersHello" holds the greeting uttered by the user. It is uninitialized in the beginning of ths cript.
- A *domain* "helloWorld" contains just one *stage* with a very simple task: obtain a greeting from the user.
- An *obtain* element linked to our only variable. The entity condition is *hypernym=45708;* which corresponds to "all kinds of 'greetings". NLUI Server will keep questioning the user until the user provides some kind of answer with a kind of "hello" in it. It can be "hi", "salute", "how are you doing", "hello", "hi", and other terms according to the lexicon.
- A *baseFinishMessage* element providing echoing not what the user said, but the 'greeting' entity inside the matching input.

Let's run the script in *NLUI Debugger*:



Script File: C:\Users\Vadim\Documents\Visual Studio 2008\Projects\Text2Knowledge\NLUIS\hello.config

Carabao Data Directory: C:\Carabao\lingua\

g'day mate.

This is your greeting: "g'day"

System: Well, just say a greeting.
User: g'day mate.
System: This is your greeting: "g'day"

Building and Debugging NLUI Server Scripts

Please refer to the supplied *NLUI Server Script Configuration Reference* for in-depth reference on the NLUI script schema.

In order to shorten the development cycle, we provide a GUI tool named NLUI Debugger (shown in the illustration above) to test the scripts without the need to deploy them in a server environment, and gain insight into what is happening on every step. Load your script (not service!) configuration, modify the Carabao lexicon folder if needed, select the active language, and start your testing.

- It is much easier to create and edit the scripts with an XML editor which supports IntelliSense based on XSD schemas, such as the XML Editor built-in into Microsoft Visual Studio: http://msdn.microsoft.com/en-us/library/ms255811.aspx (but, of course, there are numerous alternatives). Use *nluis.xsd* as the schema.

Building Multilingual Scripts

As the scripts understand language using Carabao runtime, they all understand all languages in the active Carabao database. However, the prompts themselves must be customized per language. (Note that you may supply several message templates to the same message, and then they are randomized to create an illusion of human-like interaction.)

If the messages are not created for every languages aimed to be supported, then NLUI Server falls back to the default language every time. This option may be acceptable for testing, but should not be used in production.

Supported languages are explicitly specified in the *languageMap* attribute of *service* element in the script configuration.